



University  
of Glasgow

<https://theses.gla.ac.uk/>

Theses Digitisation:

<https://www.gla.ac.uk/myglasgow/research/enlighten/theses/digitisation/>

This is a digitised version of the original print thesis.

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study,  
without prior permission or charge

This work cannot be reproduced or quoted extensively from without first  
obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any  
format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author,  
title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk/>  
[research-enlighten@glasgow.ac.uk](mailto:research-enlighten@glasgow.ac.uk)

# Comparison and Visualisation of Taxonomic Hierarchies

by

**Eilidh J. Grant**

A thesis submitted to the  
Faculty of Information and Mathematical Sciences  
at the University of Glasgow  
for the degree of  
Master of Science

September 2004

© Eilidh J. Grant 2004

ProQuest Number: 10753968

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10753968

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 – 1346

GLASGOW  
UNIVERSITY  
LIBRARY

## **Acknowledgements**

I would like to thank my supervisors Ela Hunt and Rod Page, and also Nadia Anwar for testing the software.

Many thanks to Fred, Nix, Jean-Alain, Andy, Katie, Kirsty and my family for all their support.

Special thanks to Tim.

## **Abstract**

There is no one single agreed taxonomy of species. This causes problems when searching across databases as searching under taxa with the same name may not have the same meaning if different taxonomies have been used in the databases. There is therefore a need to be able to find out whether taxa in different taxonomies share the same meaning or classify some species differently.

This project presents software for addressing the problem of meaningfully comparing taxonomies. A novel algorithm for merging two hierarchical classifications and finding the differences between them is described.

We have successfully developed a tool incorporating the comparison algorithm and a visualisation of the results. This has been tested and evaluated to show that it does provide the user with a better understanding of the differences between taxonomies, and should therefore be useful as a component in future tools for data interpretation.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Taxonomy . . . . .	1
1.2	Visualisation . . . . .	2
1.3	Other applications . . . . .	3
1.4	Thesis structure . . . . .	3
<b>2</b>	<b>Thesis Statement</b>	<b>4</b>
<b>3</b>	<b>Motivation</b>	<b>5</b>
3.1	Sample Problem . . . . .	5
<b>4</b>	<b>Background</b>	<b>7</b>
4.1	Current tools . . . . .	7
4.1.1	Existing Software . . . . .	8
4.2	Algorithms . . . . .	9
4.2.1	MoReTax . . . . .	9
4.2.2	ATreeGrep . . . . .	12
4.2.3	Archiving Scientific Data . . . . .	13
4.3	Interface Design . . . . .	14
4.3.1	Prometheus . . . . .	14
4.3.2	Treebolic . . . . .	15
4.3.3	SpaceTree . . . . .	15

<b>5</b>	<b>Requirements Capture</b>	<b>19</b>
5.1	Users . . . . .	19
5.2	Requirements . . . . .	19
5.2.1	Functional Requirements . . . . .	19
5.2.2	Non-Functional Requirements . . . . .	20
<b>6</b>	<b>Algorithm</b>	<b>21</b>
6.1	Sample Problem . . . . .	21
6.1.1	Definitions . . . . .	26
6.2	Naive algorithm . . . . .	26
6.3	Algorithm . . . . .	27
6.3.1	Merge Trees . . . . .	27
6.3.1.1	Algorithm . . . . .	28
6.3.2	List Unmatched Nodes . . . . .	29
6.4	Improved Algorithm . . . . .	31
6.4.1	Further work . . . . .	31
6.5	Summary of Algorithm . . . . .	32
<b>7</b>	<b>Materials and Methods</b>	<b>33</b>
7.1	Implementation . . . . .	33
7.1.1	Java . . . . .	33
7.1.2	Handling XML Input Data . . . . .	33
7.1.3	Class diagram . . . . .	34
7.2	Results . . . . .	34
7.3	XML Data format . . . . .	37
7.3.1	TaxonTree.dtd - the Input DTD . . . . .	37
7.3.2	Sample Input Files . . . . .	38
7.3.2.1	SimpleTree 1 . . . . .	38
7.3.2.2	SimpleTree 2 . . . . .	39
7.3.3	Output Data Formats . . . . .	40



7.3.3.1	SpaceTree . . . . .	40
7.3.3.2	Treebolic . . . . .	40
<b>8</b>	<b>Visualisation</b>	<b>44</b>
8.1	Visualisation Using JTree . . . . .	44
8.2	SpaceTree . . . . .	46
8.3	Treebolic . . . . .	46
8.4	Discussion of visualisation . . . . .	49
<b>9</b>	<b>Further Work</b>	<b>52</b>
9.1	Added Functionality . . . . .	52
9.2	Algorithm . . . . .	52
9.3	Implementation . . . . .	53
9.4	Visualisation . . . . .	53
9.5	Applications in other fields . . . . .	53
<b>10</b>	<b>Conclusion</b>	<b>54</b>

# List of Figures

1.1	Linnaean classification of chimpanzee, human, lion and tiger, represented by a tree graph. . . . .	2
3.1	Two different classifications of albatrosses (family <i>Diomedidae</i> ), represented as tree graphs. The classification on the left is from Robertson and Nunn [11], the classification on the right is from the NCBI taxonomy tree. . . . .	6
4.1	NEWT query interface . . . . .	7
4.2	NEWT results browser . . . . .	8
4.3	Two different classifications of albatrosses (family <i>Diomedidae</i> ). Tree 1 on the left is from Robertson and Nunn[11]; tree 2 on the right is from the NCBI taxonomy. . . . .	11
4.4	Example Trees . . . . .	13
4.5	Set-based comparison of Taxonomies . . . . .	16
4.6	The Treebolic applet displaying a file system. . . . .	17
4.7	SpaceTree displaying the organisational hierarchy of an oil company . . . . .	18
6.1	Two different classifications of albatrosses (family <i>Diomedidae</i> ), represented as tree graphs. The classification on the left is from Robertson and Nunn [11], the classification on the right is from the NCBI taxonomy tree. . . . .	22
6.2	The two albatross classifications as above, with the species which are differently classified highlighted in red, and the non-equivalent <i>Diomedea</i> node highlighted in bold. . . . .	23

6.3	The two albatross classifications as above, with the species which occur only in one tree highlighted in green. . . . .	24
6.4	The two albatross classifications as above, with the nodes which have an equivalent node in both trees highlighted in bold. . . . .	25
6.5	Two example trees to be compared. . . . .	28
6.6	The trees merged together with the nodes labelled with which tree they originated from. The nodes which are 'unmatched', i.e. only belong to one tree are highlighted. . . . .	29
6.7	The list of unmatched nodes in alphabetical order. The nodes are coloured green if they appear once in the list and red if they occur twice. The red nodes have pointers to the corresponding node with the same name. . . . .	30
6.8	The merged tree with final coloured nodes . . . . .	30
7.1	class diagram . . . . .	35
7.2	Data flow - Input and Output XML files . . . . .	38
7.3	Simple trees 1 and 2, merged and displayed in SpaceTree . . . . .	41
7.4	Simple trees 1 and 2, merged and displayed in Treebolic . . . . .	43
8.1	The trees representing the classifications by NCBI and by Robertson and Nunn of the <i>Diomedidae</i> family, displayed in Java JTree. . . . .	45
8.2	A web page incorporating some of the improvements suggested by the users, including a key. . . . .	47
8.3	The merged tree for the classifications by NCBI and by Robertson and Nunn of the <i>Diomedidae</i> family, displayed in SpaceTree. . . . .	48
8.4	The merged tree for the family <i>Diomedidae</i> family, displayed in SpaceTree as above. The red nodes have been searched for and are highlighted. . . . .	48
8.5	The merged tree for the classifications by NCBI and by Robertson and Nunn of the <i>Diomedidae</i> family, displayed in the Treebolic applet. . . . .	49

# List of Tables

4.1	MoReTax comparison of albatross classifications . . . . .	10
7.1	Time taken for algorithm to run on the test data sets . . . . .	36
8.1	A Comparison of tree viewers. . . . .	50

# Chapter 1

## Introduction

In this project we aim to produce software that will allow the user to compare different classifications of species. Species are named under the Linnaean classification system. This taxonomy groups species into hierarchies, for example, humans have the species name *Homo sapiens* and are classed as members of the genus *Homo*, the order *Primates*, the class *Mammalia* and the kingdom *Animalia* (figure 1.1). The Linnaean classification system can be represented as a rooted tree with all of the nodes labelled. Each node has a “rank”, such as order, family, genus, or species. The NCBI Taxonomy database<sup>1</sup>[15] has 28 ranks. Taxonomies are not entirely standardised and so different databases may classify species differently[8]. The problem we have addressed in this project is how to spot where classifications are different and to bring this to the attention of the biologist so that they can decide how to resolve the issue. This is a special case of the “ontology matching” problem. We worked with Professor Rod Page, an evolutionary biologist with an interest in taxonomies.

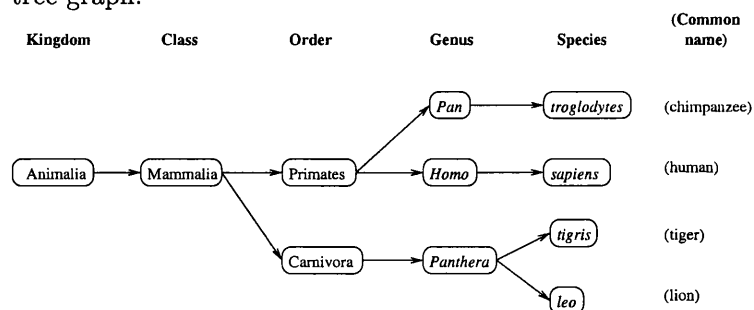
### 1.1 Taxonomy

Taxonomy has the apparently simple aim of giving every species a unique name and classifying these species in a hierarchical structure that represents their relatedness. It is

---

<sup>1</sup><http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Taxonomy>

Figure 1.1: Linnaean classification of chimpanzee, human, lion and tiger, represented by a tree graph.



important to have this standard naming system so that scientists can have meaningful discussions about species based on a common understanding. This issue of standardised naming is similar to the use of ontologies to standardise biological terminology[2]. An ontology is the common vocabulary in which shared knowledge is represented[4].

Taxonomies may differ because taxonomists disagree on how to classify species. For example it is not always obvious whether a species is one species with two different populations or actually two different species, or a species may have features of two different taxonomic groups: one taxonomist may use certain features to classify the species under one taxa; another taxonomist may decide other features of the species are more significant and classify it under a different taxa. This issue arises because taxonomies are a human construct. The plant, animals and bacteria that are alive today have evolved from common ancestors that diverged more recently the more closely related the species are. The phylogenetic relationship of species describes the evolutionary relationships between them. The phylogeny of species is constantly revised as new data, for example molecular data, becomes available. Taxonomies are more stable and the same species names will be used in different interpretations of the tree of life (phylogenies).

## 1.2 Visualisation

Visualisation can be an important aid to understanding. In this project several approaches to visualising more than one hierarchical structure at a time are discussed. For example,

viewing the trees side by side, merging the trees into one graph or using a set-based visualisation.

### **1.3 Other applications**

The problem of comparing two similar trees is also applicable when studying the history of a file system or changes to an XML document. Additions to and deletions from the tree could be of interest. When comparing taxonomies it is important to know if something has moved to a different branch of the tree and at which branching this occurred. This will be applicable in ontologies where it is important to know if a word is being used with a different meaning in different ontologies/ versions.

### **1.4 Thesis structure**

A sample problem in Chapter 3 provides the reader with context. In Chapter 4 the software that is currently used by our users is described and related algorithms and visualisations that have relevance to the problem are discussed. The results of the requirements capture are in Chapter 5. The algorithm that has been designed to solve the problem is in Chapter 6. Details of the implementation are provided in Chapter 7. The visualisation of the taxonomy tree comparison is discussed in Chapter 8 and further work in Chapter 9.

## Chapter 2

# Thesis Statement

I aim to design an algorithm which compares two taxonomies and finds taxa that are: only classified under one taxonomy; classified in the same way in both taxonomies; classified differently in each taxonomy and the points at which the different classifications diverge. I will argue that it is possible to visualise the results of the algorithm in a way that allows the user to understand the comparison of the two taxonomies. I assert that it is possible to achieve interoperability with other programs. The validity of these statements will be tested in collaboration with phylogeny researchers.



## Chapter 3

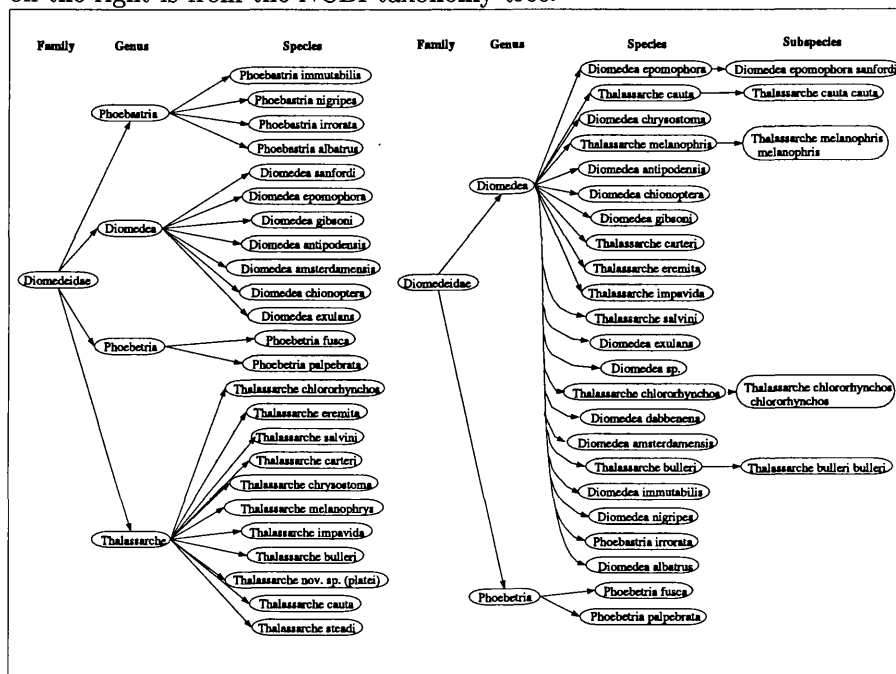
# Motivation

### 3.1 Sample Problem

We examine here the problem of comparing two different albatross classifications: the NCBI classification and the Robertson and Nunn classification (Figure 6.1). The NCBI classification divides the family *Diomedidae* into two genera, *Diomedea* and *Phoebastria*, whereas the Robertson and Nunn classification recognised four genera; *Diomedea*, *Phoebastria*, *Phoebastria* and *Thalassarche*. The species which are classified under *Diomedea*, *Phoebastria* and *Thalassarche* in the Robertson and Nunn classification are all classified together under the genus *Diomedea* in the NCBI classification. This means that a search for species belonging to the genus *Diomedea* under each of the two classifications has a different meaning and will return different species.

The aim of this project is to write an algorithm that can find those taxa which have different meanings in different taxonomies. The differences between the taxonomies should also be represented in a meaningful way.

Figure 3.1: Two different classifications of albatrosses (family *Diomedidae*), represented as tree graphs. The classification on the left is from Robertson and Nunn [11], the classification on the right is from the NCBI taxonomy tree.



## Chapter 4


# Background

### 4.1 Current tools

There are tools available for accessing taxonomies, such as the NCBI taxonomy interface<sup>1</sup> and NEWT[6], a new taxonomy portal to the SWISS-PROT protein sequence knowledge-base, see Figures 4.1 and 4.2. These do provide some information on synonyms in other databases but these rely on someone curating the database. The software described in this project would help with this data curation task or could be used directly by users of the database to view the relationships between different data sets directly.

<sup>1</sup><http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Taxonomy>

Figure 4.1: NEWT query interface<sup>2</sup>



To query NEWT:

Enter text:

match: ☒ complete word  
☐ substring

query: ☒ official names and official synonyms  
☐ all names and all synonyms

or Taxonomy ID:

Figure 4.2: NEWT results browser

Diomedea			
Lineage	Tax ID	37068	
<ul style="list-style-type: none"> <li>Eukaryota</li> <li>Metazoa</li> <li>Chordata</li> <li>Crinata</li> <li>Vertebrata</li> <li>Euteleostomi</li> <li>Archaeopteryx</li> <li>Aves</li> <li>Neognathae</li> <li>Procellariiformes</li> <li>Diomededidae</li> </ul>	Scientific name	Diomedea	
	other NCBI synonyms	Phoebastria Thalasarche	
	Rank	genus	
	Number of Swiss-Prot entries	1	
	Number of TrEMBL entries	20	

Taxonomy navigation	
Up taxonomy tree	Down taxonomy tree
<ul style="list-style-type: none"> <li>• <i>Diomedea</i></li> </ul>	<ul style="list-style-type: none"> <li>• <i>Diomedea nigripes</i></li> <li>• <i>Diomedea amsterdamensis</i></li> <li>• <i>Diomedea antipodensis</i></li> <li>• <i>Diomedea chinensis</i></li> <li>• <i>Diomedea chrysostoma</i></li> <li>• <i>Diomedea dabryanensis</i></li> <li>• <i>Diomedea eximiodora</i></li> <li>• <i>Diomedea exulans</i></li> <li>• <i>Diomedea gilchristi</i></li> <li>• <i>Diomedea immutabilis</i></li> <li>• <i>Diomedea melanoleuca</i></li> <li>• <i>Diomedea nigripes</i></li> <li>• <i>Diomedea nigripes</i></li> <li>• <i>Diomedea sp.</i></li> <li>• <i>Phaethon rubricauda</i></li> <li>• <i>Thalasarche buchanani</i></li> <li>• <i>Thalasarche castroii</i></li> <li>• <i>Thalasarche caudata</i></li> <li>• <i>Thalasarche chlorostictus</i></li> <li>• <i>Thalasarche crenatipennis</i></li> <li>• <i>Thalasarche melanocephala</i></li> <li>• <i>Thalasarche salvini</i></li> </ul>

#### 4.1.1 Existing Software

The Glasgow Taxonomic Name Server<sup>3</sup> is a web site created by Rod Page. This allows users to search for a classification or species and returns the associated identity numbers from the GenBank<sup>4</sup> and ITIS (Integrated Taxonomic Information System)<sup>5</sup> databases, and other data sources such as the Robertson and Nunn classification [11]. At present a node in the classification tree of one database is considered to be equivalent to a node in another database if they share the same name and rank, although we have built a more sophisticated system to highlight when this is not the case. The Glasgow Taxonomic Name Server is home to a database of taxonomic classifications and can provide this data in XML format<sup>6</sup>. This is where our application will acquire its data.

<sup>3</sup><http://darwin.zoology.gla.ac.uk/~rpage/MyToL/www/index.php>

<sup>4</sup><http://www.ncbi.nlm.nih.gov/Genbank/GenbankOverview.html>

<sup>5</sup><http://www.itis.usda.gov/>

<sup>6</sup><http://www.w3.org/XML/>

## 4.2 Algorithms

Looking at similar problems in other domains will inform the design of a solution to the taxonomy comparison problem. Algorithms for searching for matches to trees based on comparing paths (Section 4.2.2) and for merging XML trees (Section 4.2.3) are described here.

The problem of comparing phylogenetic trees, although biologically relevant to taxonomists, is not relevant to the required taxonomy comparison algorithm design. In taxonomic trees all of the internal nodes are labelled, in phylogenetic trees the internal nodes are unlabelled. When comparing phylogenetic trees the tree structure must be compared to determine which internal nodes match best from each tree, when comparing taxonomic trees the labels on the nodes can be simply compared.

### 4.2.1 MoReTax

The MoReTax<sup>7</sup> system defines possible relationships between different taxonomic concepts, as defined by different taxonomists. These relationships are used to merge data about these taxa, so that it can be accessed despite being referenced differently.

The following basic relationships from set theory are used to describe the relationship between two taxonomic concepts T1 and T2:

- R1.**  $T1 \equiv T2$  T1 and T2 are *congruent*. Every member of T1 is a member of T2 and *vice versa*.
- R2.**  $T1 \subset T2$  T1 is *included in* T2. Every member of T1 is a member of T2. Some members of T2 are not members of T1.
- R3.**  $T1 \supset T2$  T1 *includes* T2. Every member of T2 is a member of T1. Some members of T1 are not members of T2.
- R4.**  $T1 \oplus T2$  T1 and T2 *overlap* each other. Some members of T1 are members of T2. Some members of T1 are not members of T2. Some members of T2 are not members

---

<sup>7</sup><http://www.bgbm.org/biodivinf/projects/moretax>

Table 4.1: MoReTax comparison of albatross classifications

$Diomedeidae\ 1 \oplus Diomedeidae\ 2$	$Diomedeidae\ 1 \oplus Diomedea\ 2$	$Diomedeidae\ 1 \supset Phoebeiria\ 2$
$Phoebastria\ 1 \oplus Diomedeidae\ 2$	$Phoebastria\ 1 \oplus Diomedea\ 2$	$Phoebastria\ 1 \not\equiv Phoebeiria\ 2$
$Diomedea\ 1 \oplus Diomedeidae\ 2$	$Diomedea\ 1 \oplus Diomedea\ 2$	$Diomedea\ 1 \not\equiv Phoebeiria\ 2$
$Phoebeiria\ 1 \subset Diomedeidae\ 2$	$Phoebeiria\ 1 \not\equiv Diomedea\ 2$	$Phoebeiria\ 1 \equiv Phoebeiria\ 2$
$Thalassarche\ 1 \oplus Diomedeidae\ 2$	$Thalassarche\ 1 \oplus Diomedea\ 2$	$Thalassarche\ 1 \not\equiv Phoebeiria\ 2$

of T1.

**R5.**  $T1 \not\equiv T2$  T1 and T2 *exclude* each other. None of the members of T1 are members of T2.

The first and last relationships above are simple cases in which the taxonomic concepts are either identical or entirely unrelated. The relationships *included in* and *includes* would be of interest when deciding whether searching under one taxonomic concept in a database would yield the same results as searching under another taxonomic concept. For example if T1 is included in T2, then searching under T2 would produce at least all of the results that searching under T1 would. However if T1 and T2 overlap each other, searching under T2 may or may not produce all of the results that searching under T1 would.

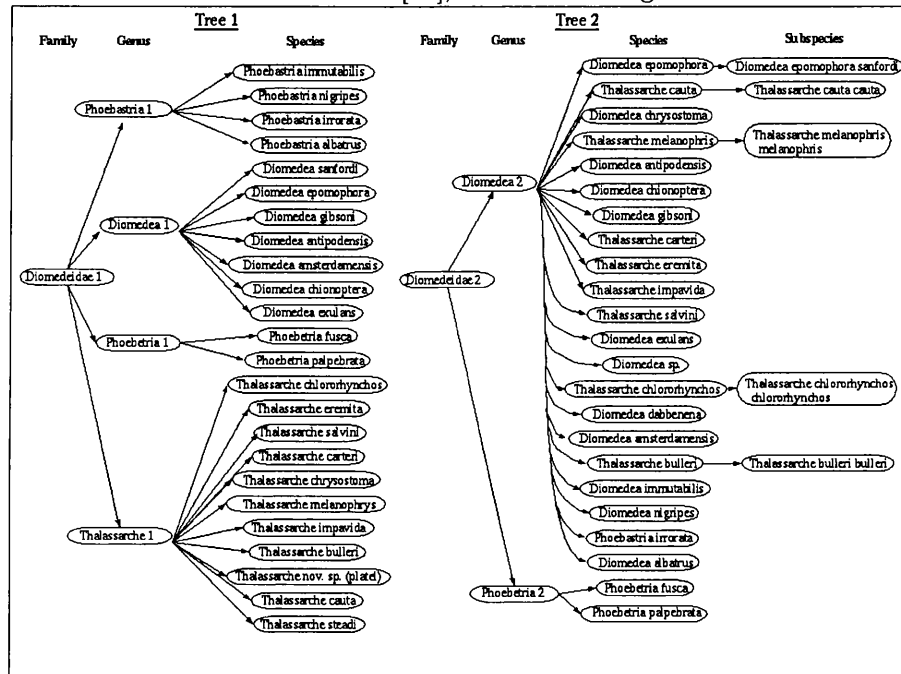
Taxonomic concepts will often be equivalent to each other, in that they do not classify any species differently, but contain some extra species that are not classified under the other concept. This leads to taxonomic concepts that would be expected to be *included in* another concept being defined as *overlapping* the other concept instead, as shown in the example below.

The albatross classification comparison previously described in the motivation chapter is used as an example here (Figure 6.1). The family and genus level nodes from the left hand tree are compared with each of the family and genus level nodes on the right hand tree, and their relationships, as defined by MoReTax are shown in Table 4.1.

It would be expected that *Diomedea 1* would be included in *Diomedeidae 2*, but because *Diomedea 1* contains an extra species (*Diomedea sanfordii*) that is not classified in tree 2, these two taxa are defined as overlapping.

The MoReTax system does not readily solve the problem described in chapter 3. Many

Figure 4.3: Two different classifications of albatrosses (family *Diomededidae*). Tree 1 on the left is from Robertson and Nunn[11]; tree 2 on the right is from the NCBI taxonomy.



taxa will be described as overlapping, and it takes some interpretation to discover whether the taxa are equivalent apart from species that are only described in one of the classifications, or not. In the table above, looking only at comparisons of genus with genus, it can be seen that *Diomedea 2* overlaps with *Phoebastria 1*, *Diomedea 1* and *Thalassarche 1*. This shows that *Diomedea1* and *Diomedea 2* are not equivalent but this is not immediately apparent.

The MoReTax system also consists of a database linking taxonomic and other data from various websites, and a web based user interface. There are a set of inference rules that can be used to decide whether data on a taxon in one classification applies to a taxon in another classification, based on the basic relationship between those taxa. However, because species that are only classified under one of the trees are used when defining the basic relationship, information is lost and taxa that could be considered to be included in another taxa, are defined as overlapping and it is then uncertain whether data can be transferred from one taxa to another.

#### 4.2.2 ATreeGrep

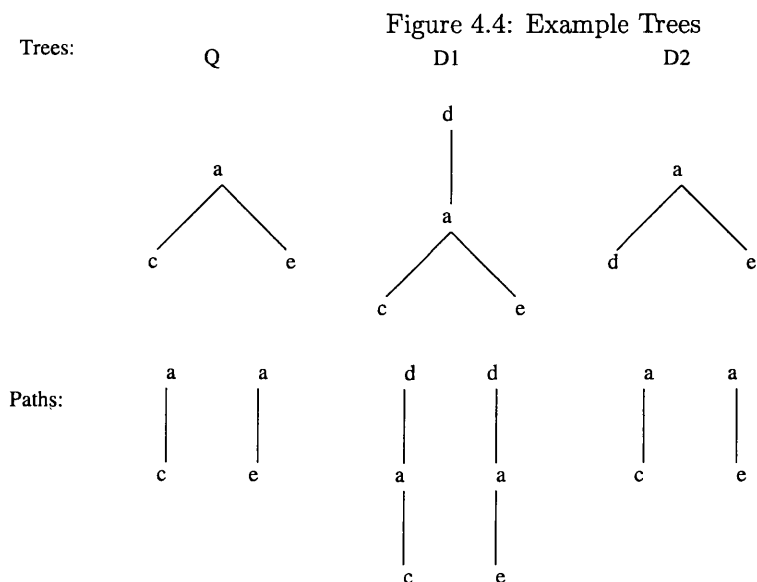
**Aim** The aim of Shasha *et al.*'s ATreeGrep [12] program is to search for approximate matches to a query tree ( $Q$ ) within a database of unordered labelled trees ( $D$ ).

**Algorithm** The algorithm used in ATreeGrep first indexes every root-to-leaf path in all of the trees in  $D$ , and then searches for matches to the root-to-leaf paths of the query tree  $Q$  using the index.

There is one root-to-leaf path for every leaf in a tree (Figure 4.4). In a labelled tree this path can be represented as a string made up of the labels of the nodes in the order they appear in the root-to-leaf path. These strings are concatenated with a delimiter such as \$ between each string. Every suffix of this string is then indexed in a suffix array.

In the on-line search phase of the algorithm, each root-to-leaf path of the query tree is searched for in the suffix array database. The time complexity of this search is  $O(q^2 \log S)$ , where  $q$  is the number of leaves in the query tree and  $S$  is the size of the suffix array. It





takes  $q \log S$  to search the suffix array once and the maximum possible number of paths in  $Q$  to be searched is  $q$ . If all of the paths in  $Q$  are found in one of the trees in the database, then this is an exact match. Approximate matches can be found by allowing a certain number of paths not to match.

The ATreeGrep algorithm is an efficient way to search many trees for matches to a query tree, however for our problem where we are given two trees of comparable size and depth, the advantages of this algorithm are lost. The ATreeGrep algorithm does not provide a solution to the question (described in Chapter 3) of whether two nodes are equivalent. The aim of this project is to compare trees to discover if a node has been moved to a different place in the tree, and at which place this difference occurs. The ATreeGrep algorithm will only find that trees differ on a certain number of paths and will not find where these differences occur.

### 4.2.3 Archiving Scientific Data

A related problem of archiving scientific data in XML format has been addressed by Buneman et.al.[1] There are many scientific databases that are regularly updated, and all of the

previous versions must be archived, as other scientific work will be based on that data. The data is in XML format and so it naturally forms a tree structure. As part of the solution to the problem of storing all of the versions of the data, the nodes in the latest version to be added to the archive tree are compared with the nodes in the archive and then merged into one data archive tree. The nodes are timestamped with a version number - the key feature of this XML data archiving algorithm. This is similar to our work in that in both cases a comparison of XML structures is made. The X-Diff program[14] also compares XML structures in such a way that the next version can be programatically reconstructed from the previous version of an XML file. The comparison of the XML structures will not necessarily be meaningful to the users of the XML, in this case the biologists.

## 4.3 Interface Design

Visualisation software is already used in conjunction with the Glasgow Taxonomic Name Server. The tree viewers SpaceTree and Treebolic can be used to view single taxonomies and are discussed below. The comparison and visualisation of classification heirarchies has previously been investigated by Kennedy *et. al.* and their work is described in the following section.

### 4.3.1 Prometheus

Prometheus is a taxonomic database[10]. As part of this system, the visualisation of multiple taxonomic classifications has been investigated by Graham, Kennedy and Hand[3]. Their system was designed to meet the requirements of a group of plant taxonomists. These users wanted to be able to view several different classifications at once, and:

1. To track a particular genus's siblings and parents across re-organised taxonomic structures, if present.
2. To track a particular higher-level node's children across re-organised taxonomic structures, if present.

3. To compare the number of distinct levels within and across a set of taxonomic hierarchies.
4. To compare the structure of whole classifications against each other, though this was stated to be an infrequent and secondary task.

This differs from our requirements in that while the taxonomists are looking for a descriptive, historical understanding of taxonomies, our users are biologists who want to know what the equivalent of a taxon in one database is in another.

Graham et. al. investigate the use of graph-based and set-based visualisations to compare taxonomies. The graph-based visualisation shows the taxonomies combined into one directed acyclic graph. The set-based visualisation uses Tufte's concept of small multiples [13] and displays each hierarchy individually. So that all of the trees can fit on the screen at once, space is saved by not labelling the leaf nodes (species) - the name is displayed when the mouse is over that species. The taxonomies can be compared by using a brushing technique where the selected nodes in one tree are highlighted in the other trees too. This is illustrated in Figure 4.5.

#### 4.3.2 Treebolic

Treebolic<sup>8</sup> is a Java applet for displaying hierarchical data in hyperbolic (curved) space[7]. A tree is rendered in curved space so that the selected node is largest and in the centre of the display. Parent and child nodes of the selected node are rendered slightly smaller and nodes that are further away are smaller still (Figure 4.6). This gives the effect of the hierarchy being viewed through a fisheye lens. Animation is used to show a gradual change when the focus is changed.

#### 4.3.3 SpaceTree

SpaceTree[9] is another tree visualisation tool. SpaceTree addresses the issue of gaining an overview of the whole tree, whilst examining some of the details, by showing the selected

---

<sup>8</sup><http://treebolic.sourceforge.net/en/home.htm>

Figure 4.5: Set-based comparison of Taxonomies

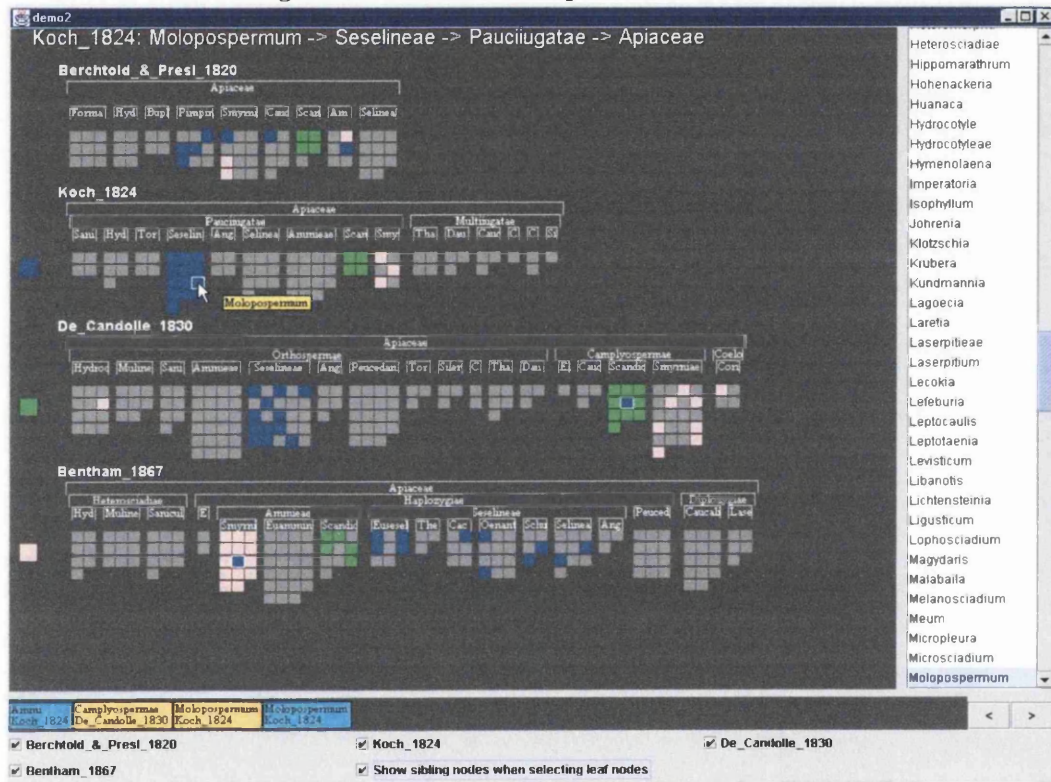
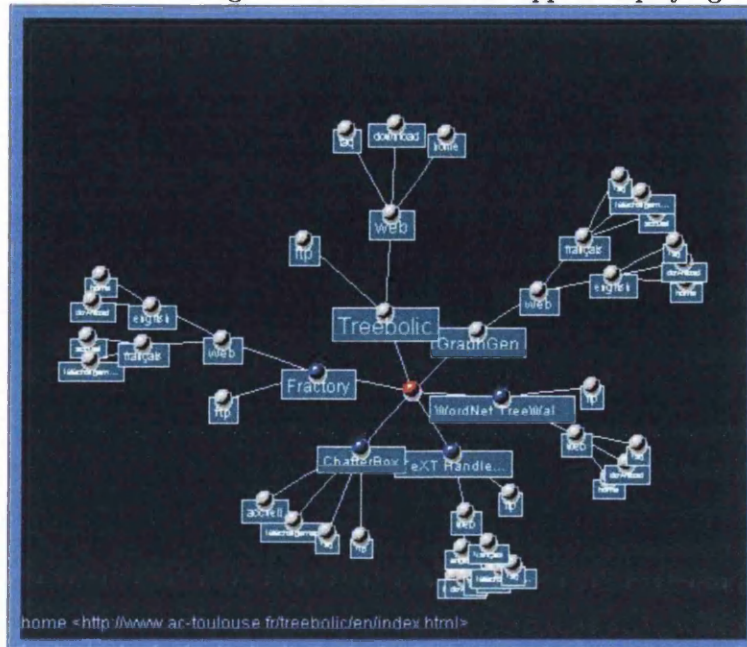


Figure 4.6: The Treebolic applet displaying a file system.



node, the path to the selected node and if possible the selected node's children in detail. Where there is not room to display nodes in detail the size (breadth and depth) of the subtree containing those nodes is represented by a triangle as illustrated in Figure 4.7. The transition from one selected node to another is animated, so that the user is not disorientated by a sudden change in representation.

Figure 4.7: SpaceTree displaying the organisational hierarchy of an oil company



## Chapter 5

# Requirements Capture

### 5.1 Users

The system will be used by people who understand taxonomies. The users of the Glasgow Taxonomic Name Server may use this product to compare different taxonomies from the Server. They are therefore likely to want to access the product from the Internet. The system may also be used by members of Rod Page's lab to decide how to annotate taxa in their database.

### 5.2 Requirements

#### 5.2.1 Functional Requirements

**F0** Enable the comparison of two taxonomic classifications, in particular highlighting the similarities and differences between the classifications.

**F1** There should be a visual display of the classifications.

**F2** It should be possible to see which species exist only in one tree.

**F3** It should be possible to see which species are in both trees and classified in the same way.

**F4** It should be possible to see which species are in both trees and classified in different ways.

**F5** For the species which are classified differently it should be possible to see where the differences occur.

**F6** It should be possible to output data from the system in formats that can be read by visualisation tools such as Treebolic (a hyperbolic tree viewer)<sup>1</sup> and SpaceTree[9] (another tree viewer), and to connect to these viewers.

### **5.2.2 Non-Functional Requirements**

**NF1** The system should be simple to use.

**NF2** The system should be accessible over the Internet.

**NF3** Data to be read by our software is provided in Rod Page's XML format, currently stored on his web server.

---

<sup>1</sup><http://treebolic.sourceforge.net/en/home.htm>



## Chapter 6

# Algorithm

### 6.1 Sample Problem

We examine the sample problem of comparing two different albatross classifications, presented in the motivation chapter, in more detail here. The major difference between the two classifications is that species which are classified under *Diomedea*, *Phoebastria* and *Thalassarche* in the Robertson and Nunn classification are all classified together under the genus *Diomedea* in the NCBI classification (Figure 6.1). This means that a search for species belonging to the genus *Diomedea* under each of the two classifications has a different meaning and will return different species.

The nodes labelled *Diomedea* in each of the two trees have the same name and level of classification (i.e. genus) but are not equivalent. We know this because there are some species classified under *Diomedea* in the NCBI classification which are classified under a different genus (e.g. *Thalassarche*) in the Robertson and Nunn classification (see Figure 6.2).

A node in one tree is equivalent to a node in another tree if it has the same leaf nodes (species) as descendants. In most cases there will be species which only belong to one of the classifications, these can be ignored when deciding whether two nodes are equivalent (Figure 6.3). In the example above the nodes representing the genus *Phoebastria* are equivalent in both trees because they have exactly the same species as descendants.

Figure 6.1: Two different classifications of albatrosses (family *Diomedidae*), represented as tree graphs. The classification on the left is from Robertson and Nunn [11], the classification on the right is from the NCBI taxonomy tree.

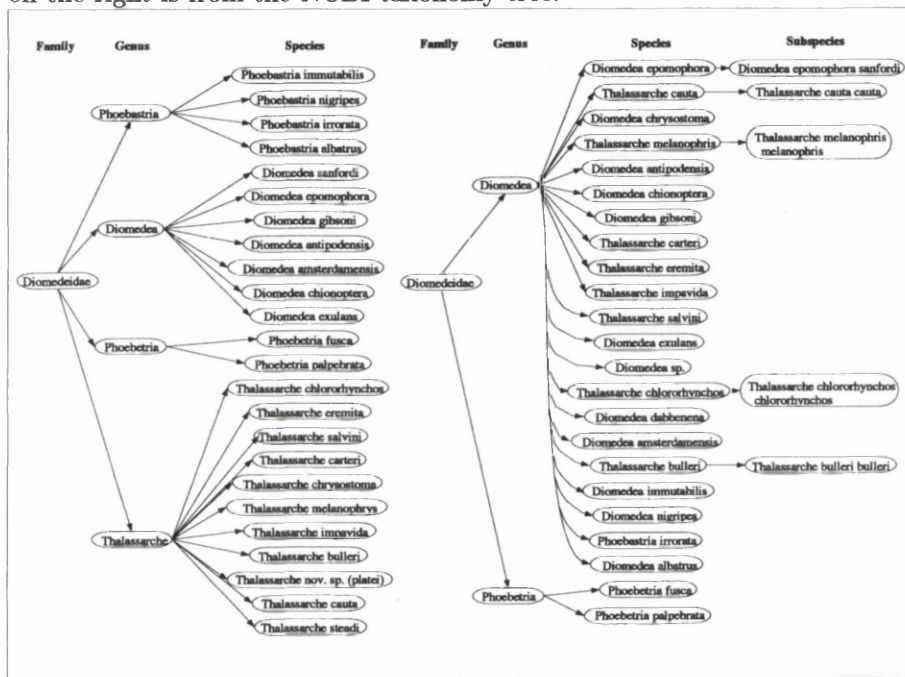


Figure 6.2: The two albatross classifications as above, with the species which are differently classified highlighted in red, and the non-equivalent *Diomedea* node highlighted in bold.

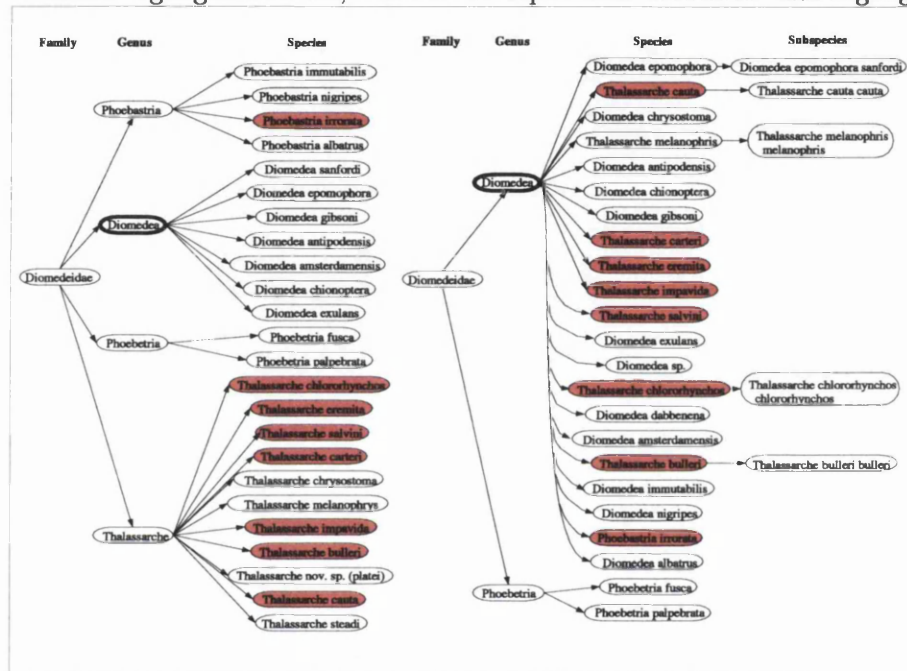


Figure 6.3: The two albatross classifications as above, with the species which occur only in one tree highlighted in green.

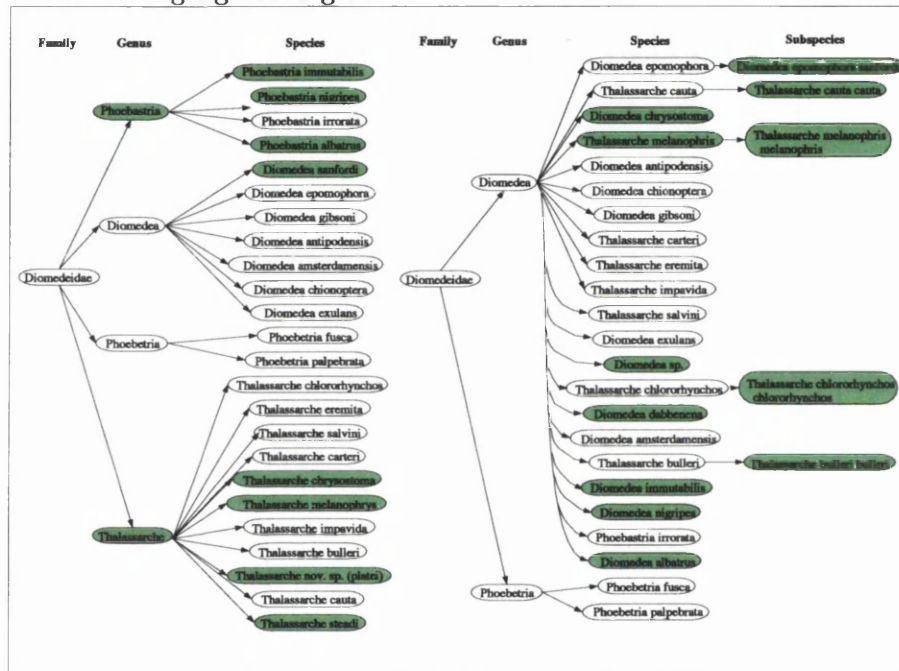
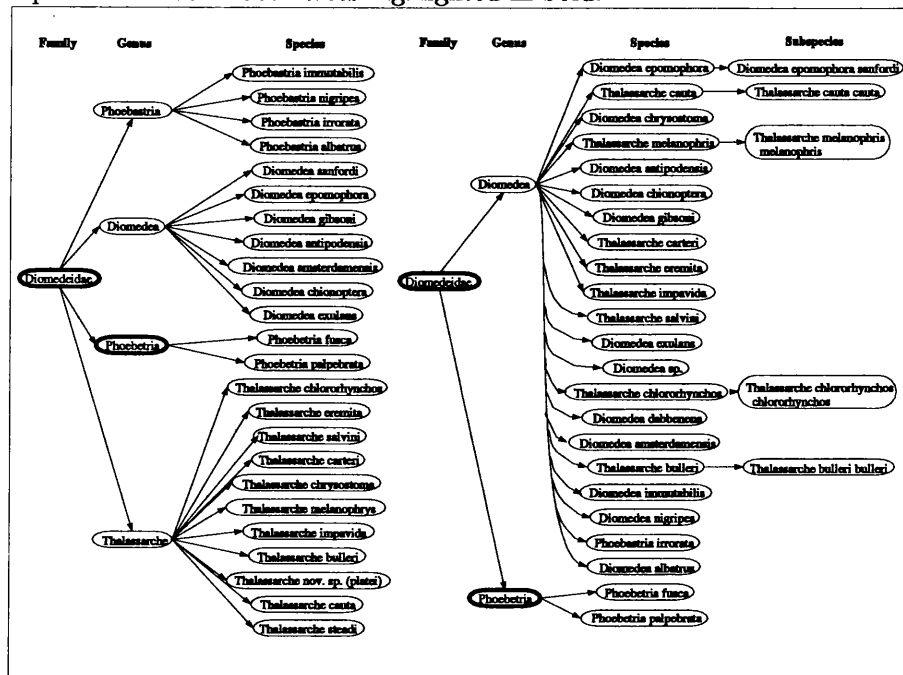


Figure 6.4: The two albatross classifications as above, with the nodes which have an equivalent node in both trees highlighted in bold.



The nodes representing the family *Diomedidae* are also equivalent, as the only species which differ between the two *Diomedidae* classifications are species which only occur in one of the classifications (Figure 6.4). It should be noted that the top nodes of the trees, in this case *Diomedidae*, will always be found to be equivalent. Without looking at higher levels of the classification we can't tell whether species that appear to occur in only one of the classifications are actually in the other classification but classified under a different node. An important difference between our approach and that of MoReTax is that we ignore nodes that don't occur in both trees when deciding if nodes are equivalent.

Species which are named differently in each classification are treated as different species in our algorithm. For example, *Phoebea immutabilis* and *Diomedea immutabilis* are the same species but named differently in the two classifications. If a lookup table of equivalent names were available then this information could be incorporated into the algorithm.

### 6.1.1 Definitions

**Matched nodes (white)** Internal or leaf nodes that appear in both trees under the same path from the root.

**Unmatched nodes** Internal or leaf nodes that do not appear in both trees under the same path from the root.

**Unique nodes (green)** Internal or leaf nodes that occur only in one tree.

**Mismatched nodes (red)** Leaf nodes (species or subspecies) that occur in both trees under different paths from the root.

**Equivalent nodes** Equivalent nodes have the same name and rank in both trees. The leaves below this node are not found anywhere else in the other tree except under the equivalent node in the other tree.

**Conflicting/ non-equivalent nodes (amber)** The internal nodes which are the point at which the paths of the mismatched nodes mismatch.

## 6.2 Naive algorithm

A naive approach could be to take each species name in one tree and search for the same species name in the other tree. If no match is found then that species is only found in one of the trees, can be labelled as such and considered no further. If there is a species with the same name in the other tree, then the path to the root for each of the species can be compared. If the path is the same then the species are classified in the same way and can be labelled as the same. If the paths between the species name and the root of each tree differ, then the species are classified differently and the species and the nodes at which they differ should be labelled as not equivalent. In effect we are comparing the full classification of every species in one classification system with the corresponding classification in the

other classification system. The complexity of this algorithm is of the order of  $O(n^2)$ . This algorithm is similar to the path matching approach used in the ATreeGrep algorithm.

In the example above, the species *Diomedea sanfordii* and *Diomedea dabbenena* are only classified under the NCBI classification and the Robertson and Nunn classification respectively. These can be labelled as only belonging to one classification and ignored for the purpose of deciding whether nodes in the two trees are equivalent. The species *Phoebetria fusca* and *Phoebetria palpebrata* are in both classifications and have the same path from the root, that is: *Diomedeidae*, *Phoebtria*, *Phoebetria fusca* / *palpebrata*. They are therefore classified in the same way under both classifications and the node *Phoebtria* is equivalent in both trees. The species *Thalassarche carteri* is classified under both classifications but the path from the root in the NCBI classification is *Diomedeidae*, *Diomedea*, *Thalassarche carteri* and the path from the root in the Robertson and Nunn classification is *Diomedeidae*, *Thalassarche*, *Thalassarche carteri*. These differ at the genus level, i.e. *Diomedea* or *Thalassarche*.

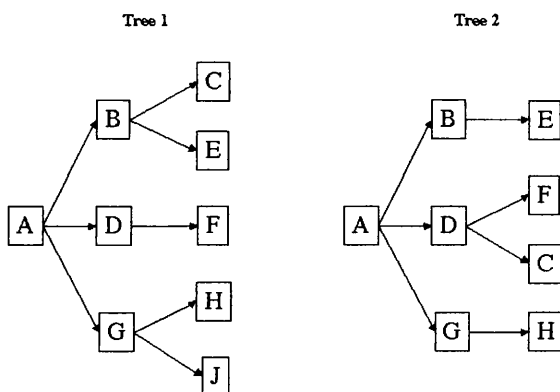
## 6.3 Algorithm

Our approach to the problem of comparing two classification trees has been to merge both of the trees into one tree and label each node as belonging to tree 1 and/or tree 2. Each node also has a colour - white, green or red, with white as the default. At the end of the algorithm the nodes which are equivalent should be coloured white, the nodes which exist only in one tree should be coloured green, the nodes which differ between trees should be coloured red and the nodes that are not equivalent should be coloured amber. We consider two small example trees in Figure 6.5. The XML files defining these trees are shown in Section 7.3.2.

### 6.3.1 Merge Trees

The first tree is read in from an XML formatted file and every node is labelled as belonging to tree 1. The second tree is then read in, and if a node with the same name at the same

Figure 6.5: Two example trees to be compared.



level exists, it is labelled as belonging to tree 2 as well as tree 1. If no node with the same name exists then a new one is created and labelled as belonging to tree 2 (figure 6.6). All nodes are white at this stage.

#### 6.3.1.1 Algorithm

The tree merging algorithm is shown below.

The XML Handler class calls the `addChild` method in the most recent parent `TaxonTreeNode`, when a new taxon is read in from the XML input file.

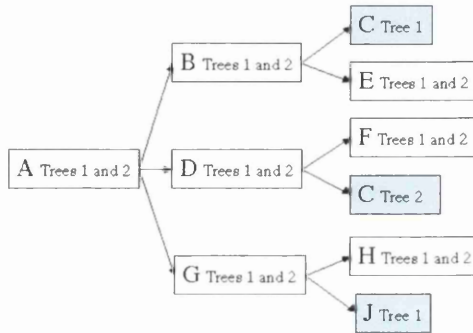
```

addChild(String name, String rank, int whichTree){
    c= getChild(n); // get child node with same name as node to be added
    if(c==null){ // if there is no matching node
        c= new taxonTreeNode(n, r); // then create and add a new node
        c.setParent(this);
        children.add(c);
    }
}

```



Figure 6.6: The trees merged together with the nodes labelled with which tree they originated from. The nodes which are 'unmatched', i.e. only belong to one tree are highlighted.



```

    }
    c.setTree(whichTree);
  }

```

### 6.3.2 List Unmatched Nodes

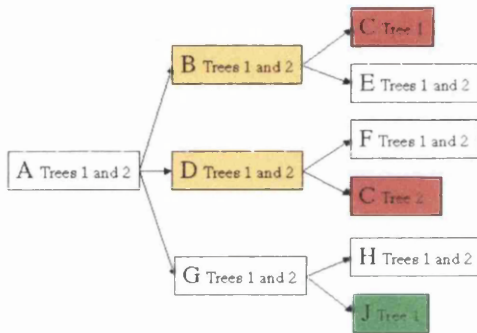
The tree is then traversed using a pre-order traversal. If a node belongs to both trees, then it is left as coloured white. If a node only belongs to one tree, then it is added to a list of unmatched nodes, sorted by name. The list of unmatched nodes is empty at the beginning of the merged tree traversal, and as nodes are added, an alphabetical ordering is maintained. When a node is added to the list of unmatched nodes, the list is searched using binary search to find the correct position for that node to be inserted.

This list of unmatched nodes is then traversed and if a node with a certain name only occurs once it is coloured green. If a node with the same name and rank occurs twice in the list it is coloured red, as it must have been classified differently in the two classifications

Figure 6.7: The list of unmatched nodes in alphabetical order. The nodes are coloured green if they appear once in the list and red if they occur twice. The red nodes have pointers to the corresponding node with the same name.



Figure 6.8: The merged tree with final coloured nodes



(figure 6.7). When nodes are labelled as red they are also set to have a pointer to the other red node with the same name. This allows the visualisation tool to access corresponding nodes with the same name without having to search the list of unmatched nodes again. The paths from the root to the red nodes can be compared and the places where they differ are nodes that are not equivalent in the two classifications. These nodes are coloured amber and a reference to the corresponding node that this node is mismatched with is stored for use in the visualisation.

This algorithm could be extended to compare any number of trees. The trees would be read into one merged tree and labelled as belonging to tree  $n$ , as above. The merged

tree would then be traversed and any nodes which did not belong to all  $n$  trees would be added to the list of unmatched nodes. In the list of unmatched nodes, any nodes which only occur once will be coloured green and nodes which occur more than once in different places will be coloured red.

## 6.4 Improved Algorithm

Our software is designed to compare two trees at present, so we adapted the above algorithm to handle two trees more efficiently. We set every node to be green by default when reading in the first tree. When the second tree is read in, if there is a corresponding node from the first tree, this will be coloured white, if not a new node will be created and coloured green. The tree can then be traversed and all of the green nodes read into a list of unmatched nodes. This list can be traversed and if a node of the same name and rank appears twice, then these nodes will be coloured red.

The algorithm can be further improved by storing the children of a node in a sorted array. This will reduce the time needed for the insertion of the second tree into the first.

### 6.4.1 Further work

The algorithm will not handle the case in which the two trees have a different number of ranks. The algorithm could also be extended to handle more than two trees and also to incorporate a lookup table of synonyms.

The list of unmatched nodes could be used further to check for spelling mistakes. Nodes that are coloured green by the algorithm could be compared using dynamic-programming string matching algorithms[5] to find nodes with similar names in the list that are only found in the opposite tree. These could be presented to the user as possible misspelled names. In the lice data set the nodes *Docophorides niethamerri* in Tree 1 and *Docophorides niethammeri* in Tree2; *Haematopinus pacocoheri* in Tree 1 and *Haematopinus phacocoheri* in Tree 2 and *Heptapsogaster minuta* in Tree 1 and *Heptapsogaster minutus* in Tree 2. The list could also be viewed by an expert as a list of candidate species that may have different

names in the two trees.

## 6.5 Summary of Algorithm

The algorithm proceeds in three steps:

1. Data from one tree is read in , labelled as belonging to Tree 1 and coloured green.
2. Data from the second tree is read in, labelled as belonging to Tree 2, and merged with Tree 1. Nodes that match in both trees are coloured white; unmatched nodes are coloured green.
3. A list of unmatched (green) nodes is made.
4. The unmatched nodes are re-labelled as unique (green) or mismatched (red) and the nodes at which the mismatched nodes don't match are labelled as conflicting (amber).

The implementation of the algorithm is described in Chapter 7.

## Chapter 7

# Materials and Methods

### 7.1 Implementation

#### 7.1.1 Java

Java<sup>1</sup> was used to implement the tree matching algorithm. The Swing library of graphical user interface classes enabled the building of an interactive tree visualisation component. It will be possible to convert the application into an Applet to run over the Internet within a web browser such as Netscape.

#### 7.1.2 Handling XML Input Data

XML<sup>2</sup> stands for extensible markup language. There are two main APIs for handling XML data - SAX(the Simple API for XML) and DOM(Document Object Model). Parsers implementing SAX read one part of the XML at a time and leave the programmer to decide what to do with the data. The DOM parser reads the whole XML document into memory in one go and makes it into a tree object representation of the data. The SAX parser was used in this project to read in the XML taxonomy data as the procedural method enabled the tree merging algorithm to insert the data from both XML files into one tree.

---

<sup>1</sup><http://java.sun.com>

<sup>2</sup><http://www.w3.org/XML/>

### 7.1.3 Class diagram

The class `TaxonTreeParser` reads in XML documents according to the specified DTD and translates all related events into `TaxonTreeHandler` events. `TaxonTreeHandlerImpl` implements the `TaxonTreeHandler` interface to handle the `TaxonTreeHandler` events. The tree traversal and the main algorithm are implemented here and in the `TaxonTreeNode` class.

The `TaxonTreeNode` class contains the data read in from the XML files, i.e. “name” and “rank”, recorded as Strings, and which classification trees the node belongs to, recorded as booleans. The colour of the node, as determined by the main algorithm, is stored in this class. Each `TaxonTreeNode` has a Vector of references to its child `TaxonTreeNodes`, its parent `TaxonTreeNode` and in the case of red and amber nodes a reference to its matching `TaxonTreeNode`. A red node will match the node in the other tree with the same name. Amber nodes signify the point at which the mismatch between two red nodes with the same name occurs. An amber node will match the node that is mismatched in the other tree. The matching node reference is used by the display so that when a node is selected in one tree, its closest match in the other tree will also be highlighted.

`TwoTreesFrame` displays the two trees with coloured nodes as described in the algorithm chapter, Chapter 6. `TreeRenderer` defines how the trees should be rendered, for example with coloured icons representing the colour of the node.

The `SpaceTree` class outputs the merged tree data in an XML format that can be displayed by the Space Tree program (Section 4.3.3), similarly, the `Treebolic` class outputs the data in an XML format that can be displayed by the Treebolic program, as described in Section 4.3.2.

## 7.2 Results

The algorithm was tested on three data sets: simple trees 1 and 2 (Section 6.3); the albatross taxonomies (Section 6.3); and also a larger data set of lice taxonomies from NCBI and Vincent Smith<sup>3</sup>. In the albatross taxonomy the algorithm correctly labelled

---

<sup>3</sup>[http://darwin.zoology.gla.ac.uk/~rpage/MyToL/www/find\\_name\\_result.php](http://darwin.zoology.gla.ac.uk/~rpage/MyToL/www/find_name_result.php)

35

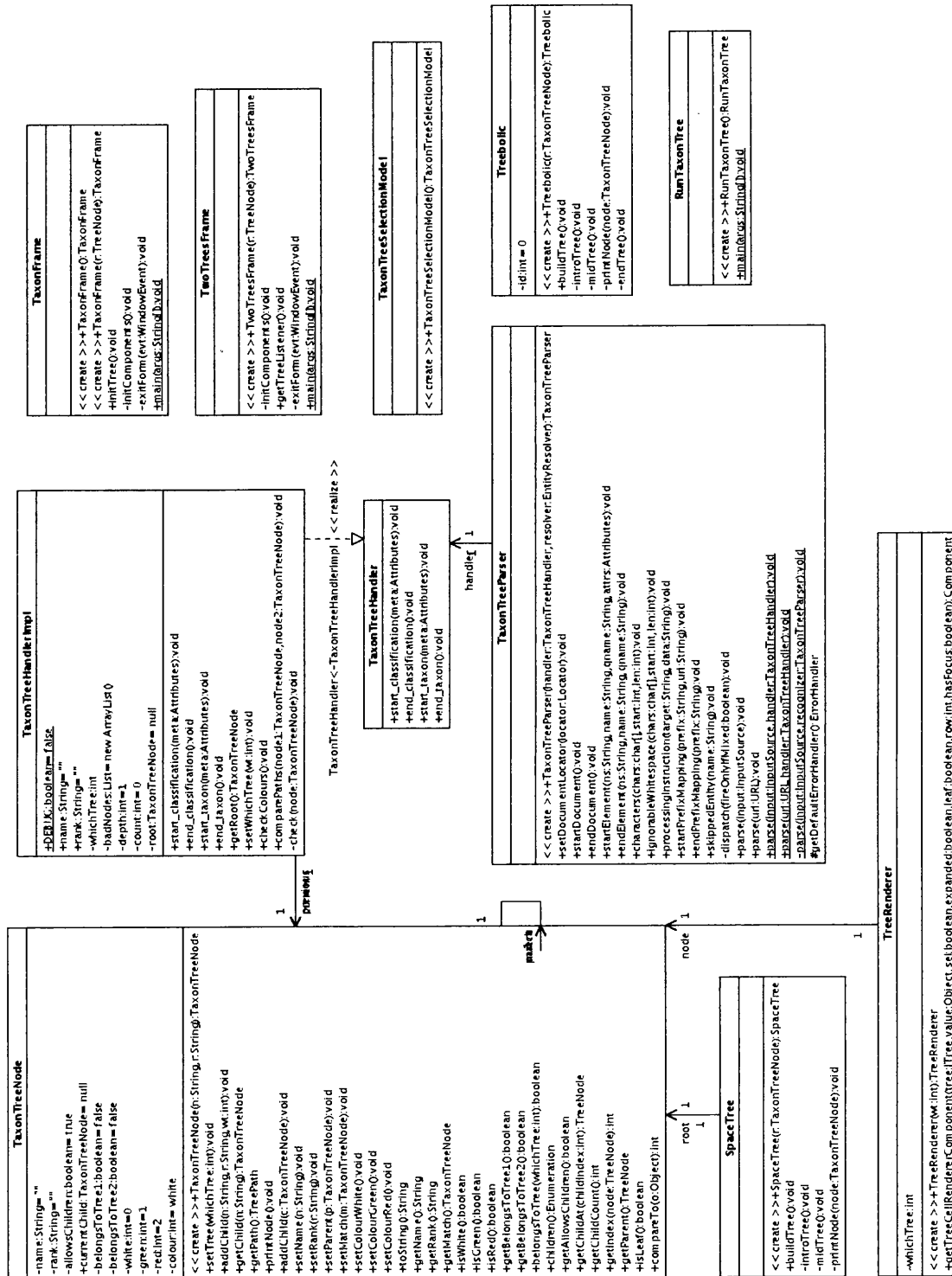


Table 7.1: Time taken for algorithm to run on the test data sets

Data set	no. of nodes in Tree 1	no. of nodes in Tree 2	no. of differing nodes	Algorithm (ms)	Visualisation - JTree (ms)	XML Output (ms)
Simple Tree1 vs Simple Tree2	9	8	3	433	5314	131
Albatross NCBI vs RobNunn	31	29	38	460	4885	178
Lice NCBI vs Vince	428	5243	5037	1603	5668	581
Simple Tree1 vs Simple Tree1	9	9	0	493	5083	99
Albatross Rob- Nunn vs Rob- Nunn	31	31	0	373	4298	137
Lice NCBI vs NCBI	428	428	0	818	5006	330
Lice Vince vs Vince	5243	5243	0	2056	4800	893

*Diomedea*, *Phoebastria* and *Thalassarche* in the Robertson and Nunn classification and *Diomedea* in the NCBI classification as not equivalent, i.e. amber. In the lice taxonomy, only two pairs of amber nodes were found. These were places where a taxon had been spelled differently in the different databases: *Rhynchophthrinia* and *Rhyncophthrinia*, and *Boopidae* and *Boopiidae*.

The times taken for the program to run the algorithm (including input), the JTree visualisation and to write the output XML files, using several data sets, are shown in Table 7.1<sup>4</sup>. The time to display JTree is constant across the test data. The time to output the SpaceTree and Treebolic XML is proportional to the size of the merged tree. The time taken to run the algorithm increases as the number of nodes in the input files increase. The algorithm would be expected to run faster when the trees to be compared match, than when the compared trees have many differences between them. In both cases the algorithm proceeds by merging the two trees into one tree, and then processing a list of unmatched

<sup>4</sup>The timings in this table are the average of 5 runs. The program was run on an iBook with an 800 MHz PowerPC G3 processor and 640 MB of memory.



nodes. If the two trees match perfectly, the list of unmatched nodes will be empty and the algorithm will finish at that point. The more mismatches there are, the longer the list of unmatched nodes will be and the longer it will take to process. To test this input files were matched with a copy of themselves. The simple tree data set actually took longer to process when both trees were identical, but this may be due to variations in timing and the small size of the data set. The algorithm did process the data faster when the two tree matched exactly (373 ms) than when there were differences between the trees (460 ms). The lice data is harder to compare because the two trees are such different sizes: 428 nodes *vs* 5243 nodes. Comparing the smaller tree (NCBI) with itself is faster than comparing the different lice trees, but comparing the large tree (Vince) with itself is slower.

## 7.3 XML Data format

XML has emerged as the *de facto* standard for data exchange between disparate systems and there are many freely available tools for parsing and manipulating data in this format. Hence, XML was used as the input format for the developed software. The Document Type Definition (DTD) and sample input and output files (Figure 7.2) are described below. Taxonomic classifications in this data format can be downloaded from the Glasgow Taxonomic Name Server web site<sup>5</sup>.

### 7.3.1 TaxonTree.dtd - the Input DTD

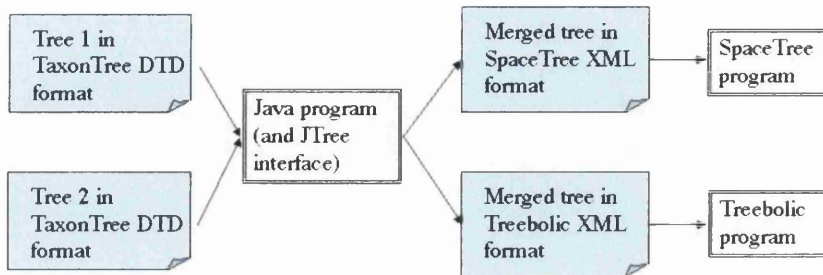
The document contains one classification, made up of many taxa. Each taxon has a name and rank (e.g. genus, species) and can contain other taxa.

```
<?xml version='1.0' encoding='UTF-8'?>
<!ELEMENT taxon (taxon)*>
<!ATTLIST taxon
    rank CDATA #IMPLIED
    name CDATA #IMPLIED
```

---

<sup>5</sup><http://darwin.zoology.gla.ac.uk/~rpage/MyToL/www/index.php>

Figure 7.2: Data flow - Input and Output XML files



>

```
<!ELEMENT classification (taxon)*>
```

### 7.3.2 Sample Input Files

We present here sample input files for the two simple trees described in Section 6.3.

#### 7.3.2.1 SimpleTree 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE classification SYSTEM "TaxonTree.dtd">
<classification>
  <taxon name="A" rank="Family">
    <taxon name="B" rank="Genus">
      <taxon name="C" rank="Species"/>
      <taxon name="E" rank="Species"/>
    </taxon>
  </taxon>
</classification>
```

```

    </taxon>
    <taxon name="D" rank="Genus">
      <taxon name="F" rank="Species"/>
    </taxon>
    <taxon name="G" rank="Genus">
      <taxon name="H" rank="Species"/>
      <taxon name="J" rank="Species"/>
    </taxon>
  </taxon>
</classification>

```

### 7.3.2.2 SimpleTree 2

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE classification SYSTEM "TaxonTree.dtd">
<classification>
  <taxon name="A" rank="Family">
    <taxon name="B" rank="Genus">
      <taxon name="E" rank="Species"/>
    </taxon>
    <taxon name="D" rank="Genus">
      <taxon name="C" rank="Species"/>
      <taxon name="F" rank="Species"/>
    </taxon>
    <taxon name="G" rank="Genus">
      <taxon name="H" rank="Species"/>
    </taxon>
  </taxon>
</classification>

```

### 7.3.3 Output Data Formats

The DTDs for XML data input for Treebolic<sup>6</sup> are available on the Internet. The merged tree XML files that are the output from the main Java program (when the two simple trees above are the input) and used as input for the SpaceTree and Treebolic programs are shown below.

#### 7.3.3.1 SpaceTree

```
<?xml version="1.0" encoding="UTF-8"?>
<node>A(white) Trees 1 and 2
  <node>B (amber) Tree 1 Tree 2
    <node>C (red) Tree 1</node>
    <node>E (white) Trees 1 and 2 </node>
  </node>
  <node>D (amber) Tree 1 Tree 2
    <node>F(white) Trees 1 and 2</node>
    <node>C (red) Tree 2</node>
  </node>
  <node>G(white) Trees 1 and 2
    <node>H(white) Trees 1 and 2</node>
    <node>J(green) Tree 1</node>
  </node>
</node>
```

The above XML code will produce the tree shown in figure 7.3 when displayed using SpaceTree.

#### 7.3.3.2 Treebolic

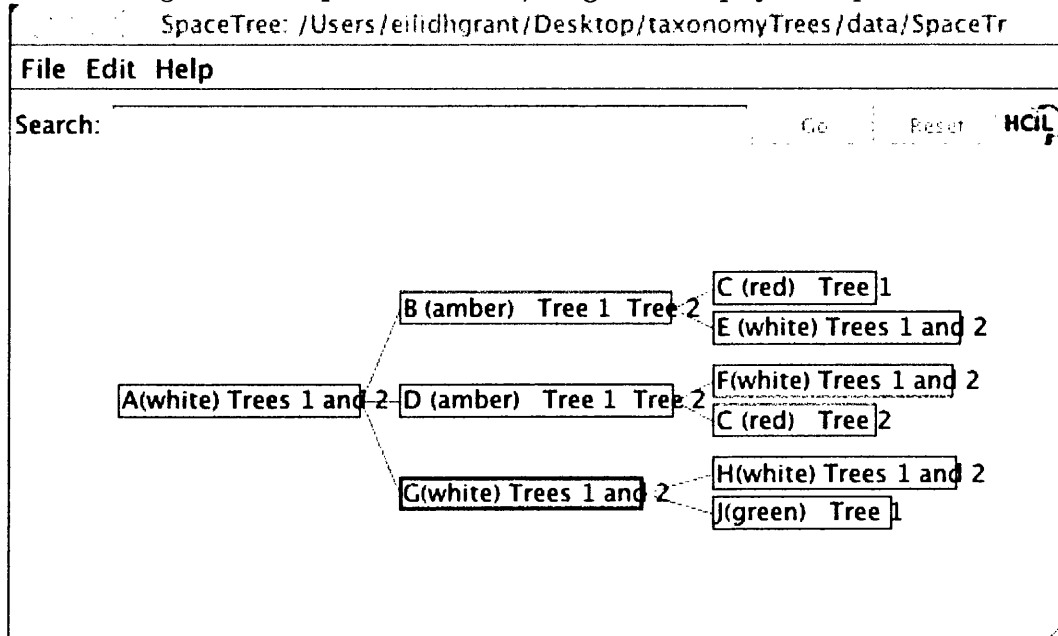
The DTD for input data for Treebolic is available on the Internet<sup>7</sup>.

---

<sup>6</sup><http://treebolic.sourceforge.net/en/dtd.htm>

<sup>7</sup><http://treebolic.sourceforge.net/en/dtd.htm>

Figure 7.3: Simple trees 1 and 2, merged and displayed in SpaceTree



```

<?xml version="1.0" encoding="UTF-8"?>
!DOCTYPE treebolic SYSTEM "Treebolic.dtd"
<treebolic>
  <tree>
    <nodes>
      <node id="1" bgcolor="000000" forecolor="FFFFFF">
        <label>A</label>
      <node id="2" bgcolor="FFCC00"> bgcolor="CCAA00">
        <label>B</label>
      <node id="3" bgcolor="FF0000">
        <label>C</label>
      </node>
      <node id="4" >
        <label>E</label>
      </node>
    
```

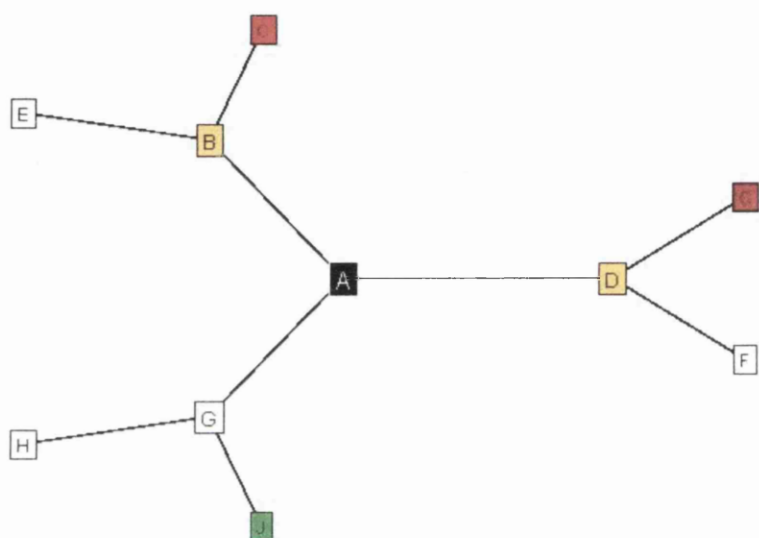
```

</node>
<node id="5" bgcolor="FFCC00"> bgcolor="CCAA00">
  <label>D</label>
  <node id="6" >
    <label>F</label>
  </node>
  <node id="7" bgcolor="CC0000">
    <label>C</label>
  </node>
</node>
<node id="8" >
  <label>G</label>
  <node id="9" >
    <label>H</label>
  </node>
  <node id="10" bgcolor="00FF00">
    <label>J</label>
  </node>
</node>
</node>
</nodes>
</tree>
</treebolic>

```

The above XML code will produce the tree shown in figure 7.4 when displayed using Treebolic.

Figure 7.4: Simple trees 1 and 2, merged and displayed in Treebolic



## Chapter 8

# Visualisation

For our visualisation of the trees we used Java Swing JTree to display the comparison of the two classifications. We also used two publicly available tree viewers - SpaceTree [9] and Treebolic <sup>1</sup> to view a merged tree representing the two classifications.

When asking users to evaluate the system we asked them to consider the following questions.

The users were asked to:

1. Find species that exist only in one tree.
2. Find which are in both trees and classified in the same way.
3. Find which species are in both trees and classified in different ways.
4. For species which have been classified differently, find where these differences occur.

### 8.1 Visualisation Using JTree

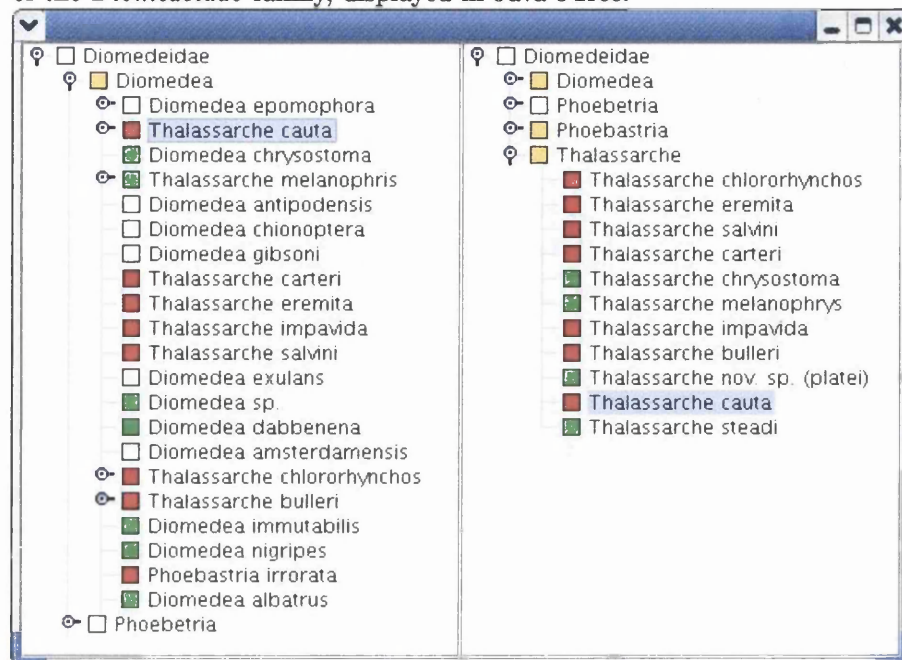
Our first visualisation tool was a Java application that shows the two classifications side by side using the the standard Java tree display JTree (Figure 8.1). In this view, when

---

<sup>1</sup><http://www.cs.umd.edu/hcil/spacetree/>



Figure 8.1: The trees representing the classifications by NCBI and by Robertson and Nunn of the *Diomedidae* family, displayed in Java JTree.



a node is selected, the corresponding node with the same name in the other classification tree will be highlighted simultaneously. Green nodes do not have a corresponding node in the other tree. Parts of the tree can be expanded or contracted by clicking on the handles to the left of expandable nodes.

We showed this visualisation to two potential users of the system: Professor Rod Page, an evolutionary biologist and Nadia Anwar, a PhD student working with Professor Page. They suggested the following improvements:

- There should be a key explaining the colouring and layout of the tree.
- When a node is chosen, the path to the root should be highlighted as well as the node. This makes it easier to see the entire classification for the node.
- There should be an option just to see the nodes that differ between the two trees, especially the conflicting nodes.

- There should also be an option to see the merged view of both classifications merged into one tree and to be able to highlight one tree at a time in this view.
- It was also suggested that the view of two trees could be laid out so that the trees face each other, the left tree with the root on the left and the right tree with the root on the right. The species nodes could then be aligned with each other in the middle. This layout would be time-consuming to implement and so has been left as a suggestion for further work.

The users also suggested some extra functionality. It would be useful for the user to be able to annotate the comparison of two trees. For example, to say that two nodes are equivalent (for example where they are spelled differently), or to comment on features of the tree. Some of this could be made interactive, allowing users to see the results of changes such as making two nodes equivalent. For example the lice comparison can be resolved if we state that *Rhynchophthrinia* and *Rhynchophthrinia*, and *Boopidae* and *Boopiidae* are equivalent.

## 8.2 SpaceTree

We output the results of our tree comparison in an XML format that could be used as an input to the SpaceTree visualisation tool (Figure 8.3). This visualisation shows the path from the root to the node clearly but does not allow you to view the whole tree at once. SpaceTree allows you to search for words in the nodes.

It was not possible to choose the colour of the nodes using the standard input format. The SpaceTree code would have to be changed to get the software to display the tree nodes in an appropriate colour. However, it was possible to search for the red nodes which showed the usefulness of being able to do this (Figure 8.4).

## 8.3 Treebolic

The results of our tree comparison were also output in an XML format that could be used as an input to the Treebolic visualisation tool (Figure 8.5). The Treebolic applet uses a

Figure 8.2: A web page incorporating some of the improvements suggested by the users, including a key.<sup>3</sup>

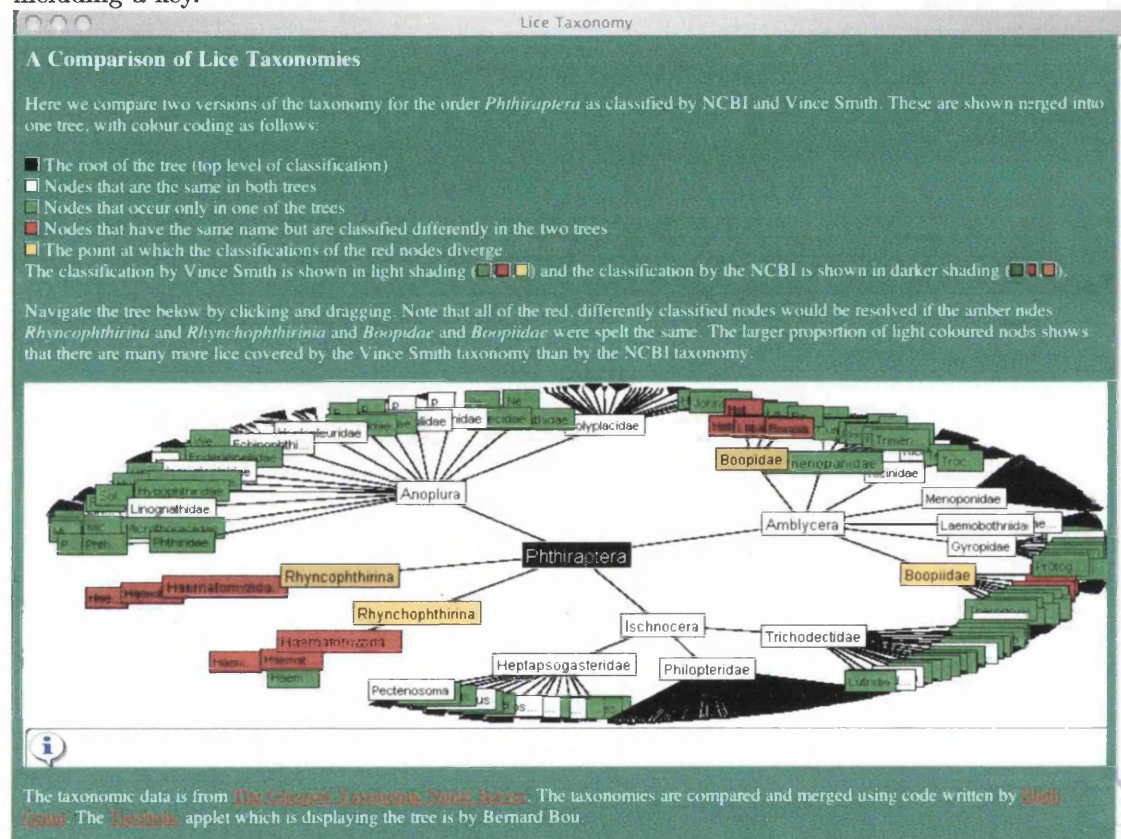


Figure 8.3: The merged tree for the classifications by NCBI and by Robertson and Nunn of the *Diomededidae* family, displayed in SpaceTree.

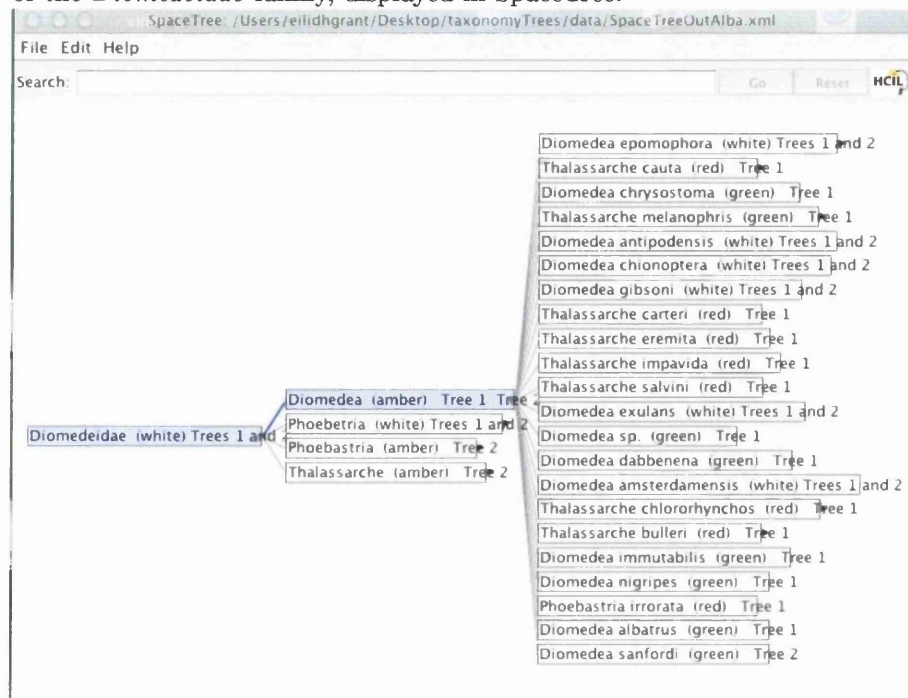


Figure 8.4: The merged tree for the family *Diomededidae* family, displayed in SpaceTree as above. The red nodes have been searched for and are highlighted.

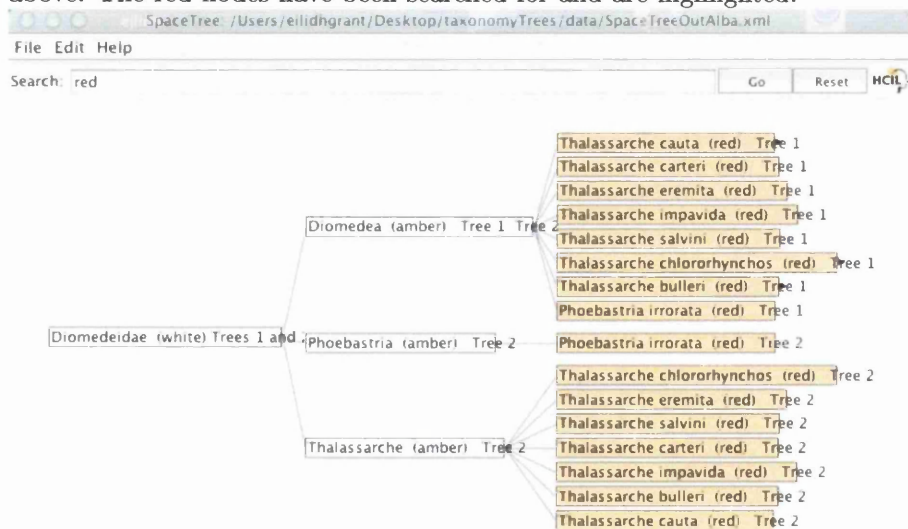
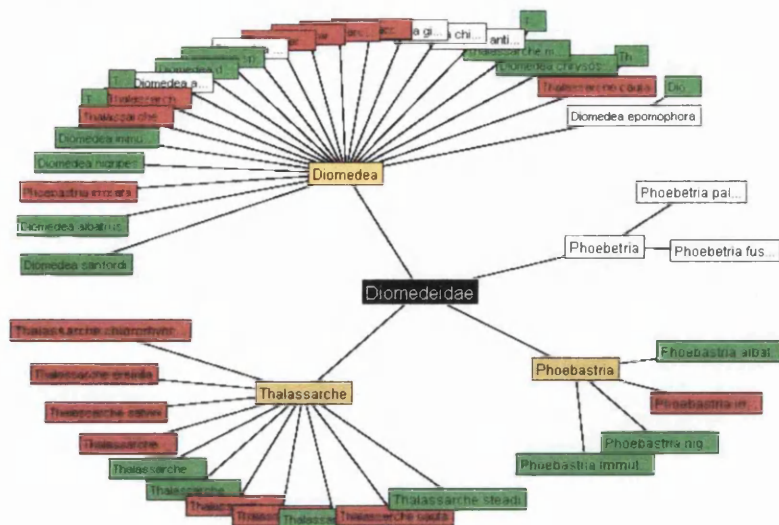


Figure 8.5: The merged tree for the classifications by NCBI and by Robertson and Nunn of the *Diomedidae* family, displayed in the Treebolic applet. <sup>5</sup>



hyperbolic visualisation technique where the tree appears as though it is on the surface of a sphere, with nodes displayed smaller and closer together the further from the centre they are. This technique allows the viewer to get a feel for the size of the tree, and to explore a lot of the tree quickly. The use of colour was particularly effective here as even when nodes were small at the edge of the screen, the colour stood out. A disadvantage of this visualisation technique is that it can be disorientating and the hierarchy of the classification was effectively lost. The Treebolic tool slows down considerably when a large tree is being viewed.

## 8.4 Discussion of visualisation

We compare the different views of the data in the table below (Table 8.1).

One of the issues when displaying hierarchical data is that of representing the overall hierarchical structure, whilst still showing details of individual nodes. In the JTree visualisation internal and leaf nodes are represented by different icons. This means that the

Table 8.1: A Comparison of tree viewers.

	Our viewer	SpaceTree	Treebolic
Display technique	like a file viewer	dynamic resizing	hyperbolic (curved space)
Layout	indented list	left to right	radial
highlights path from root	not yet	yes	no
search facility	no	yes	no
merged/ separate trees	separate (merged option to follow)	merged	merged
highlight corresponding nodes with the same name	yes	could use search facility to do this	no
legibility of display	good for comparing the two trees	easy to read path from root	reasonable
legibility of labels	good	excellent	only central labels legible

user can tell which nodes contain more nodes but has no indication of how many nodes or how many more levels of the hierarchy a node contains without expanding the node. If the expanded tree is larger than the available screen space, then JTree uses scrolling to accommodate the tree. Again this makes it hard to gain an overview of a large tree structure in JTree. SpaceTree and Treebolic both give an impression of the size and layout of the whole tree within one screen by using compression.

SpaceTree lays out the tree from left to right. Along with path highlighting this makes it easy to read the path to the selected node. The SpaceTree layout is also consistent. The JTree layout makes it a little harder to read the path to the selected node, but again the layout is always consistent. In Treebolic all of the other nodes fan out around the selected node and the orientation changes as the selected node changes. The inconsistent layout of Treebolic can be disorientating and the path from selected node to root may not be obvious.

Overall, each of the tree visualisations has its own strengths and weaknesses. JTree can show the two trees separately and when a node is selected in one tree, highlight the corresponding node in the other tree. Treebolic can provide a good overview of all of the nodes in the tree at once. SpaceTree is searchable and clearly shows the path from node

to root. It is therefore left for the user to decide which visualisation suits their needs best

- all three visualisation tools work with our algorithm.

## Chapter 9

# Further Work

### 9.1 Added Functionality

A very useful feature would be to allow users to annotate the comparison to:

- add a comment about one or several nodes.
- state that two nodes are equivalent and merge them.
- record that a node is not equivalent in the two trees.

### 9.2 Algorithm

A lookup table of synonyms could be used by the algorithm to allow for known differences when merging the trees. The algorithm should also be able to handle comparison data described above. When adding nodes to tree, the algorithm should check with annotation to see if there is an equivalent node with a different name.

The algorithm could be extended to handle more than two trees as described in Section 6.4.1.

The algorithm could also be extended to check the list of mismatched nodes for nodes that may have been spelled wrongly using dynamic programming.



### **9.3 Implementation**

As the program was written in Java it could be converted into an Applet and made available on the Glasgow Taxonomic Name Server website.

If the annotation feature was added, an extra XML format for comparison annotation storage would be necessary.

All of the algorithm is carried out in memory. For comparing very large trees it may be necessary to index the trees in a database and compare them that way.

### **9.4 Visualisation**

The layout of the view of two trees could be changed so that the trees face each other, the left tree with the root on the left and the right tree with the root on the right. The species nodes could then be aligned with each other in the middle. Further user evaluation research could be carried out to find out which visualisations are most useful to which users.

Features would need to be added to the user interface to support the added annotation functionality.

### **9.5 Applications in other fields**

This work could form the basis for XML comparison software, or software that could be used to compare ontologies.

## Chapter 10

# Conclusion

We have defined the most important features that should be highlighted when comparing taxonomic hierarchies. An algorithm has been written that compares two taxonomies by merging them and finds these features. The features are then displayed in three different visualisations using JTree, SpaceTree and Treebolic. Each visualisation has different strengths and weaknesses and so all of them are made available to allow the user to choose which suits them best.

In the thesis statement (Section 2) it was stated that the algorithm should compare two taxonomies and find taxa that are: only classified under one taxonomy; classified in the same way in both taxonomies; classified differently in each taxonomy and the points at which the different classifications diverge. The algorithm was described in Chapter 6 and tested in Chapter 7. The results of the algorithm are visualised (Chapter 8) and user testing shows that this allows the user to understand the comparison of the two taxonomies. The program is shown to interoperate with other programs (SpaceTree and Treebolic) using XML. In further work (Chapter 9), some improvements to the current implementation are suggested. It would also be interesting to use this algorithm to solve other problems such as comparing ontologies.

This project contributes a novel algorithm for comparing hierarchies and visualisations of the comparison, allowing biologists to easily see the differences between two taxonomies.

# Bibliography

- [1] Peter Buneman, Sanjeev Khanna, Keishi Tajima, and Wang-Chiew Tan. Archiving Scientific Data. *ACM Transactions on Database Systems*, 29:2–42, 2004.
- [2] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
- [3] M. Graham, J. B. Kennedy, and C. Hand. A Comparison of Set-Based and Graph-Based Visualisations of Overlapping Classification Hierarchies. In *Proceedings of the working conference on Advanced Visual Interfaces*, pages 41–50, 2000.
- [4] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5:199–220, 1993.
- [5] Dan Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.
- [6] I. Q. H. Phan et al. NEWT, a new taxonomy portal. *Nucleic Acids Research*, 31(13).
- [7] J. Lamping and R. Rao. The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies. In S. K. Card, J. D. MacKinlay, and B. Shneiderman, editors, *Readings in Information Visualisation Using Vision To Think*, pages 382–408. Morgan Kaufmann, 1999.
- [8] R.D.M. Page. Phyloinformatics: Towards a Phylogenetic Database. In J. Wang, editor, *Data Mining in Bioinformatics*. 2004.

- [9] Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proceedings of IEEE Symposium on Information Visualization*, pages 57–64, 2002.
- [10] M.R. Pullan et al. The Prometheus Taxonomic Model: a practical approach to representing multiple taxonomies. *Taxon*, 49:55–75, 2000.
- [11] C.J.R. Robertson and G.B. Nunn. Towards a new taxonomy for albatrosses. In G. Robertson and R. Gales, editors, *Albatross Biology and Conservation*, pages 13–19. Surrey Beatty and Sons, Chipping Norton, 1997.
- [12] D. Shasha, J. T. Wang, H. Shan, and Zhang K. ATreeGrep: Approximate Searching in Unordered Trees. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 89–98. IEEE Computer Society, 2002.
- [13] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1992.
- [14] Yuan Wang. X-Diff: An Effective Change Detection Algorithm for XML Documents. In *ICDE*, 2003.
- [15] D.L. Wheeler et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 28:10–14, 2000.

